Tensor decompositions and their applications

Lecture 1: Introduction

Nick Vannieuwenhoven (KU Leuven)

1. About this Masterclass (15')

2. Why tensor decompositions and their applications? (25')

3. Low-rank matrix decompositions* (35')

4. Application: analysis of energy output of solar panels (20')

5. Generalizing the SVD (10')

6. Conclusions

ANY QUESTIONS?
UH OH, I HAVE A QUESTION.
SHOULD I ASK IT? WHAT IF HE ALREADY ANSWERED IT? IS IT A DUMB QUESTION? AM I GOING TO EMBARRASS MYSELF IN FRONT OF EVERYONE?
MAYBE I SHOULD WAIT UNTIL AFTER THE TALK. DO I REALLY NEED TO ASK IT IN PUBLIC? BUT WHAT IF I'M NOT THE ONLY ONE? SHOULD I DO IT???
JORGE CHAM © 2013
YES, CECILIA?
CAN I TAKE THE LEFT OVER COOKIES HOME?
WWW.PHDCOMICS.COM

Overview

1. About this Masterclass (15')
2. Why tensor decompositions and their applications? (25')
3. Low-rank matrix decompositions* (35')
4. Application: analysis of energy output of solar panels (20')
5. Generalizing the SVD (10')
6. Conclusions

About me

Dr. Nick Vannieuwenhoven

Obtained PhD in 2015 at KU Leuven on the *tensor rank decomposition*. Spent 7 months with G. Ottaviani in Florence, Italy.

From 2015–2021 I was a FWO Postdoctoral Fellow on the *geometry of tensor decompositions*. Spent 6 months with C. Beltrán in Santander, Spain.

In 2019 became Assistant Professor of *Numerical methods and data science* at KU Leuven. Spent 18 months locked down in Leuven.

Research topics:

- Tensor and their decompositions,
- applied geometry of tensor decompositions,
- numerical analysis, and
- mathematics of data science.

About you

I assume there is some diversity in background:

- pure mathematics?
- applied mathematics / mathematical engineering?
- computer science?
- data science / machine learning / artificial intelligence?
- other engineering?

About the lectures

The plan of the lectures is as follows:

Date	Contents
Monday 8/11 11:30	Introduction
Wednesday 10/11 14:30	Tucker decomposition
Friday 12/11 9:30	Tensor trains decomposition
Monday 15/11 11:30	Tensor rank decomposition I
Wednesday 17/11 14:30	Tensor rank decomposition II

About the learning goals

In this course, we will begin to formulate answers to the following questions:

1. What's the deal with matrix decompositions?
2. What are tensors?
3. What's this tensor network notation?
4. What is a Tucker / tensor trains / tensor rank decomposition?
5. What can you do with these decompositions? What are the use cases?
6. How do you compute these decompositions?
7. How can you approximate a tensor by a low-rank decomposition?

What I cannot treat unfortunately are more global questions about the geometry of the sets of tensors with certain decompositions.

Numerical mathematics

The focus is on **numerical computing**, as opposed to symbolic computing.

In numerical computing computations are carried out using (IEEE standard double-precision) **floating-point arithmetic**. In this system, the only representable numbers are

$$\pm\left(b_1 2^{-1} + b_2 2^{-2} + \cdots + b_n 2^{-n}\right) \cdot 2^k \subset \mathbb{Q},$$

where $b_i \in \{0,1\}$, $n = 53$ and $-1021 \le k \le 1024$.

Every computation in this system resulting in a non-representable number is replaced by the nearest representable number. For the elementary operations $\circ \in \{+,-,\cdot,/\}$, we have

$$\mathrm{fl}(a \circ b) = (a \circ b)(1+\delta), \text{ where } |\delta| \le 1.1 \cdot 10^{-16}.$$

The main advantage of floating-point over symbolic computations is their **much greater speed** and **fixed memory consumption**. However, since these computations are not exact, a **different mindset** is required.

In numerical mathematics the goal is twofold:

1. Find an **approximation** of the exact quantity of interest; and
2. quantify how close this approximation is.

Software tools

There are two main **toolboxes in Matlab** for working with tensors:

1. Tensor Toolbox v3.2, developed by Kolda et al.
2. Tensorlab v3, developed by De Lathauwer et al.

The Tensor Toolbox supports the three main decompositions we will study in this Masterclass. Tensorlab has great support for the tensor rank decomposition.

ANY QUESTIONS?
UH OH, I HAVE A QUESTION.
SHOULD I ASK IT? WHAT IF HE ALREADY ANSWERED IT? IS IT A DUMB QUESTION? AM I GOING TO EMBARRASS MYSELF IN FRONT OF EVERYONE?
MAYBE I SHOULD WAIT UNTIL AFTER THE TALK. DO I REALLY NEED TO ASK IT IN PUBLIC? BUT WHAT IF I'M NOT THE ONLY ONE? SHOULD I DO IT???
YES, CECILIA?
CAN I TAKE THE LEFT OVER COOKIES HOME?
JORGE CHAM © 2013
WWW.PHDCOMICS.COM

Overview

1. About this Masterclass (15')

2. **Why tensor decompositions and their applications? (25')**

3. Low-rank matrix decompositions* (35')

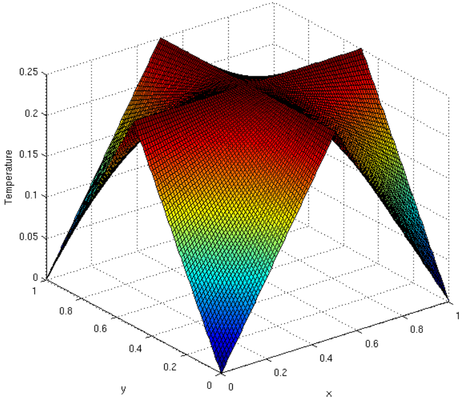
4. Application: analysis of energy output of solar panels (20')

5. Generalizing the SVD (10')

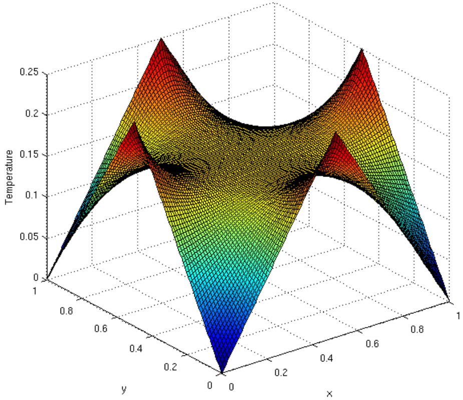
6. Conclusions

An explosion of multidimensional data

(1) Solutions of (parameterized) partial differential equations

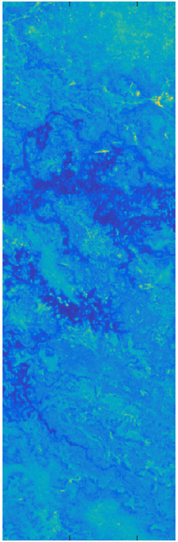


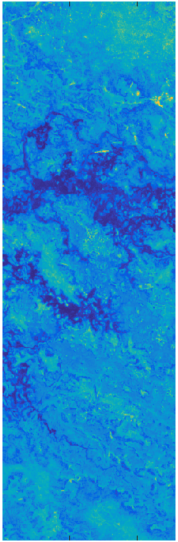
$t = 0$

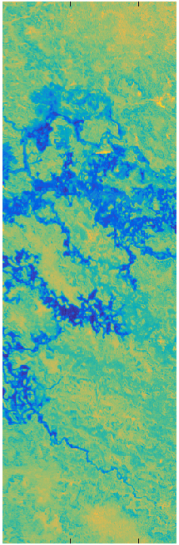


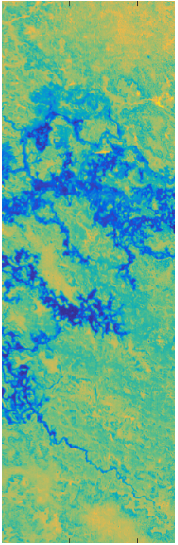
$t = 1$

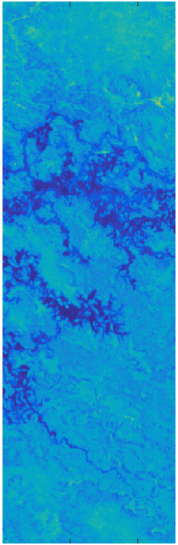
(2) hyperspectral imaging


band 5

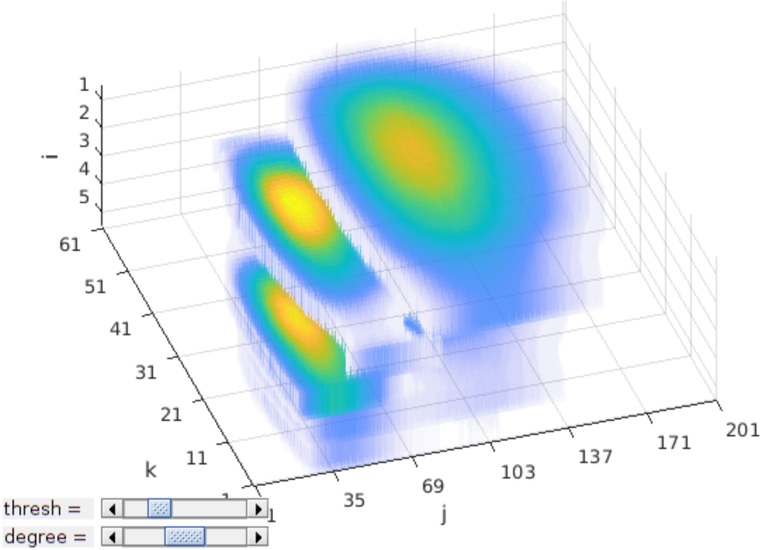

band 25


band 50

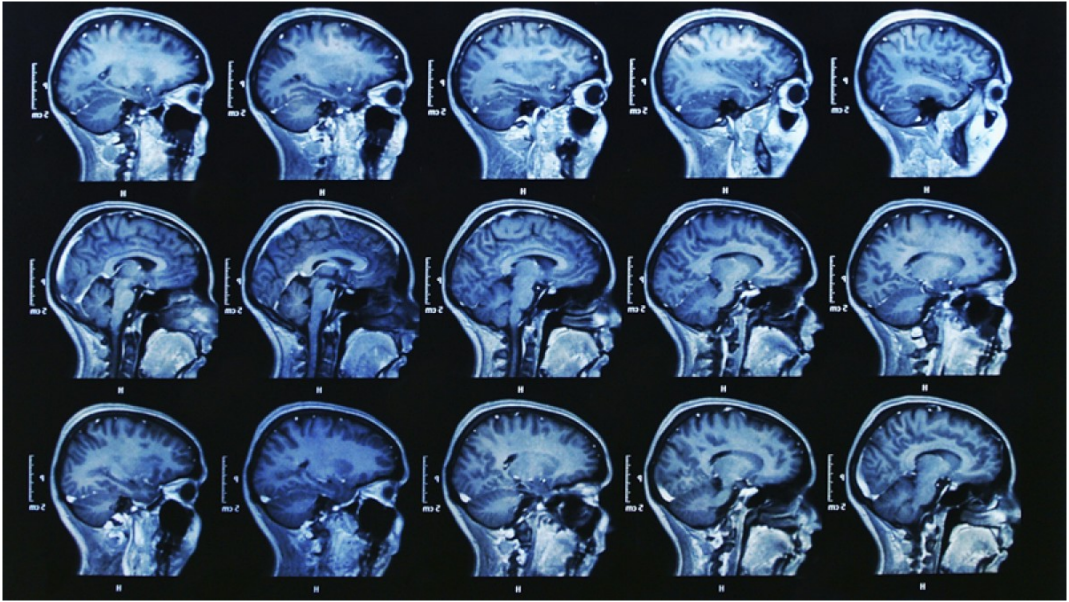

band 75


band 100

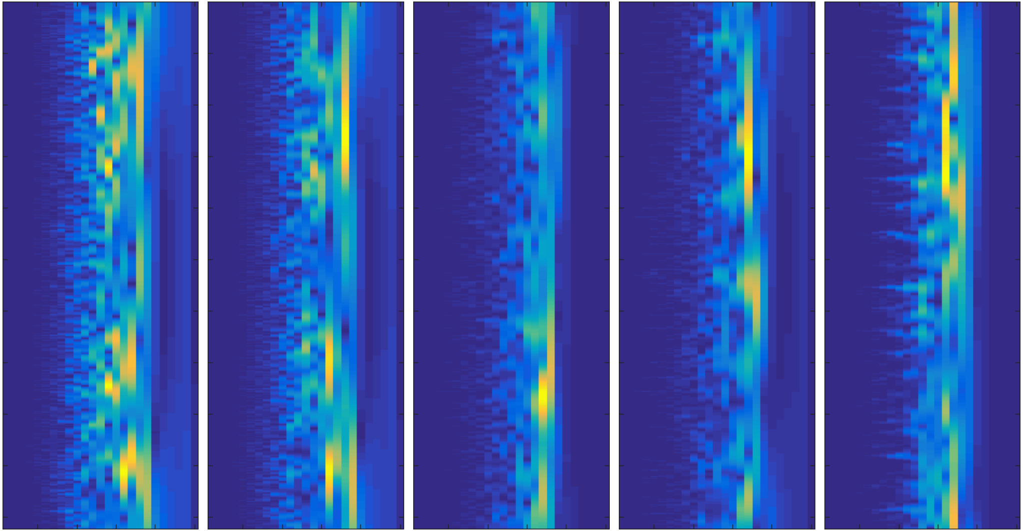
(3) fluorescence spectroscopy



(4) MRI data



(5) EEG data in the time-frequency domain



channel 1 channel 2 channel 3 channel 4 channel 5

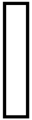
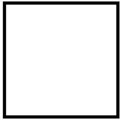
The main **challenges** associated with such data are:

- data quality (incomplete, noise, outliers),
- interpretability,
- visualization,
- analysis processing costs (time and space), and
- storage costs.

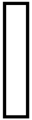
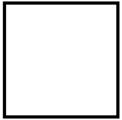
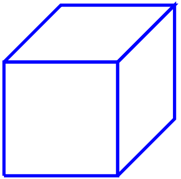
For data that varies in only 2 directions, **numerical linear algebra** offers powerful, efficient, and practical (linear) tools for

- data cleaning,
- data analysis, and
- dimensionality reduction.

Linear algebra deals with ...

$d = 0$	scalar	$a =$
$d = 1$	vector	$\mathrm{a} =$ 
$d = 2$	matrix	$A =$ 

Multilinear algebra deals with ...

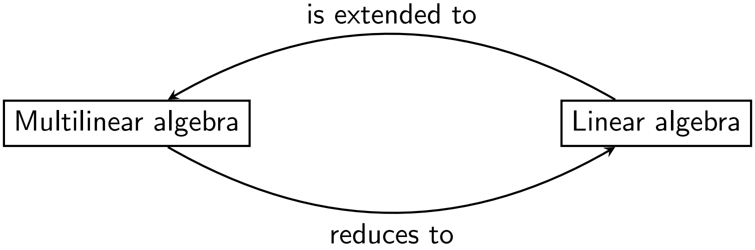
$d = 0$	scalar	$a =$
$d = 1$	vector	$\mathrm{a} =$ 
$d = 2$	matrix	$A =$ 
$d \geq 3$	tensor	$\mathcal{A} =$ 

Numerical multilinear algebra

This course will introduce you to the field of **numerical multilinear algebra**, which studies:

- the elementary multilinear objects, namely **tensors**;
- **factorizations** or **decompositions** of tensors;
- **algorithms** involving tensors;
- **sensitivity** of tensor decomposition; and
- **applications** involving tensors.

There is a natural symbiosis between (multi)linear algebra:



is extended to

Multilinear algebra

Linear algebra

reduces to

Hence, unconsciously, the developments in numerical multilinear algebra have been shaped by the same dominant ideas in **numerical linear algebra**, namely

1. **decompositions** or factorizations; and
2. **exploiting structure**.

1. Matrix decompositions

Stewart (2000) lists some benefits of the **decompositional approach to matrix computations**:

- A decomposition may solve many problems.
- A decomposition, which is generally expensive to compute, might be reused to solve new problems involving the original data.
- Many decompositions can be updated, sometimes with great savings in computation.

Given the success of framework in the context of matrices, the present focus on **tensor decompositions** is a logical extension.

Some examples of **matrix decompositions** are:

- singular value decomposition
- eigenvalue decompositions
- Schur decompositions
- Jordan decompositions
- polar decompositions
- CS decompositions
- QR-decompositions
- LU-decompositions
- interpolative decompositions

$$A = USV^T$$

$$A = VDV^{-1}$$

$$A = QUQ^{-1}$$

$$A = PJP^{-1}$$

$$A = UP$$

$$A = CS$$

$$A = QR$$

$$A = LU$$

$$A = CUR$$

In fact, many of these are special cases of **generalized Cartan decompositions**, see Edelman and Jeong (2021).

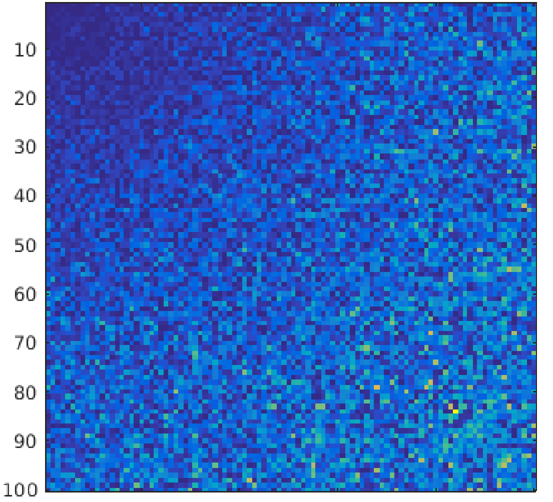
2. Exploiting structure

A second main concept is **exploiting useful matrix structures**, yielding more efficient and often more stable algorithms.

For example, discretizations (finite elements, differences or volumes) of partial differential equations (PDEs) result in huge linear systems with millions of unknowns. Storing such a system as a **dense matrix** would have a tremendous cost: a dense $10^6 \times 10^6$ matrix requires $8 \cdot 10^{12}$ bytes or 8000 gigabytes of memory![1] Fortunately, discretizations of PDEs result in **sparse** linear systems, wherein most coefficients are zero.

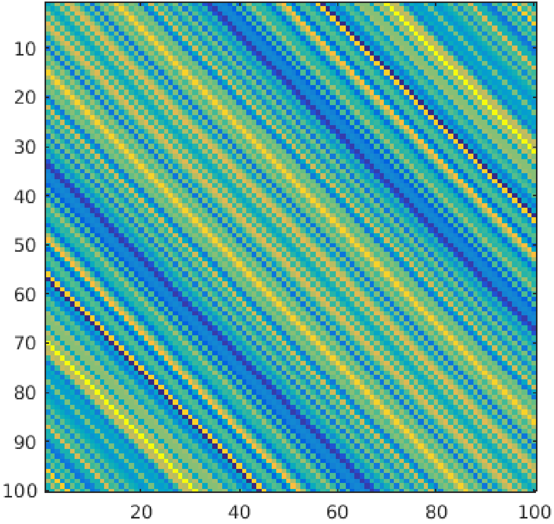
[1] That's about 30 typical 2017 laptops worth of storage.

Some well-known examples of exploitable matrix structures are:



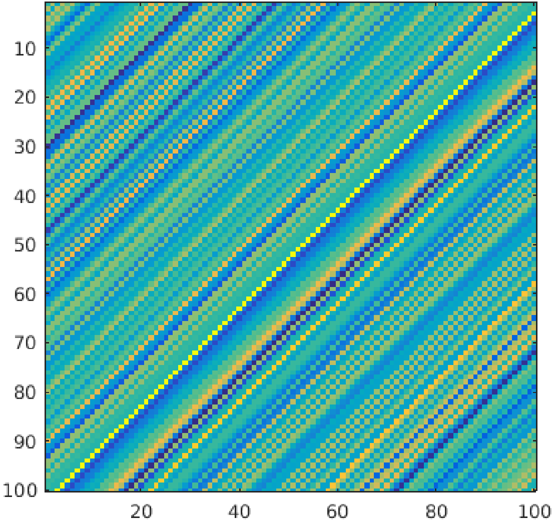
Symmetry

Some well-known examples of exploitable matrix structures are:



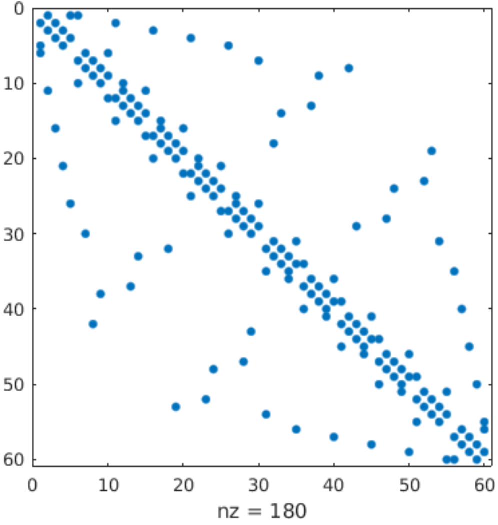
Toeplitz

Some well-known examples of exploitable matrix structures are:



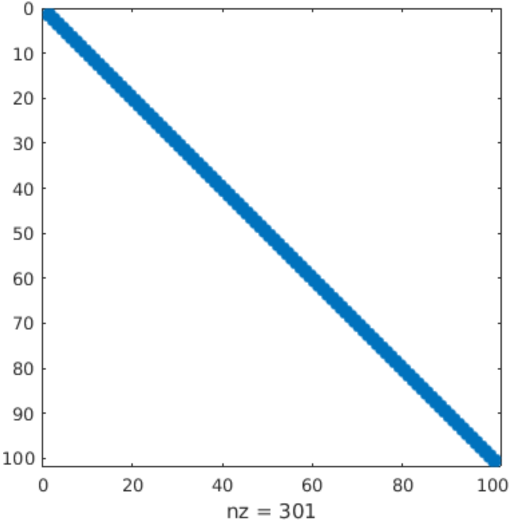
Hankel

Some well-known examples of exploitable matrix structures are:



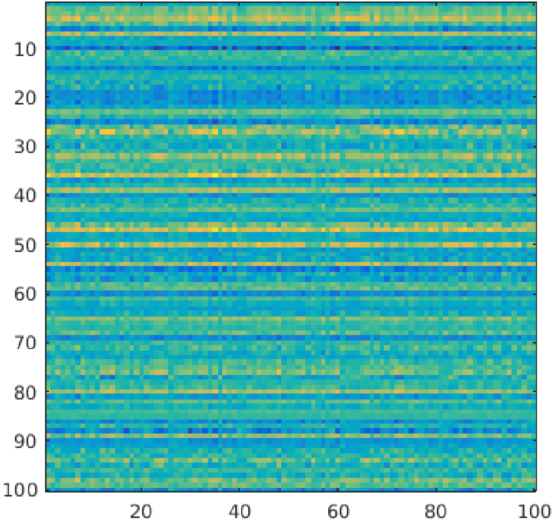
Sparsity

Some well-known examples of exploitable matrix structures are:



Banded

Some well-known examples of exploitable matrix structures are:



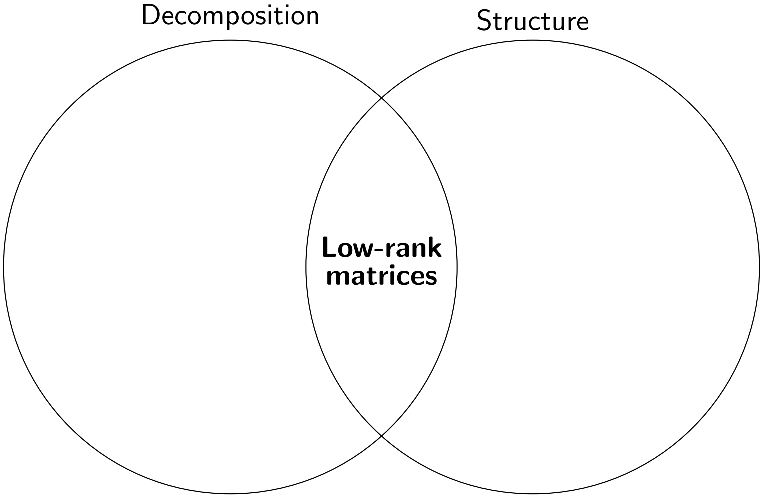
Low rank

ANY QUESTIONS?
UH OH, I HAVE A QUESTION.
SHOULD I ASK IT? WHAT IF HE ALREADY ANSWERED IT? IS IT A DUMB QUESTION? AM I GOING TO EMBARRASS MYSELF IN FRONT OF EVERYONE?
MAYBE I SHOULD WAIT UNTIL AFTER THE TALK. DO I REALLY NEED TO ASK IT IN PUBLIC? BUT WHAT IF I'M NOT THE ONLY ONE? SHOULD I DO IT???
YES, CECILIA?
CAN I TAKE THE LEFT OVER COOKIES HOME?
JORGE CHAM © 2013
WWW.PHDCOMICS.COM

Overview

1. About this Masterclass (15')
2. Why tensor decompositions and their applications? (25')
3. Low-rank matrix decompositions* (35')
4. Application: analysis of energy output of solar panels (20')
5. Generalizing the SVD (10')
6. Conclusions

Low-rank decomposition



A matrix M has a **low-rank structure** if it admits a **decomposition** as

$$M = AB^T = \sum_{i=1}^{r} \mathrm{a}_i \mathrm{b}_i^T := \sum_{i=1}^{r} \mathrm{a}_i \otimes \mathrm{b}_i$$

with r *"small"* relative to m and n, and where

$$A = \begin{bmatrix} \mathrm{a}_1 & \cdots & \mathrm{a}_r \end{bmatrix} \in \mathbb{R}^{m \times r} \text{ and } B = \begin{bmatrix} \mathrm{b}_1 & \cdots & \mathrm{b}_r \end{bmatrix} \in \mathbb{R}^{n \times r}$$

each have linearly independent columns.

The singular value decomposition

Arguably the most fundamental rank-revealing matrix decomposition is the **singular value decomposition (SVD)**. It says that every matrix can be represented as a diagonal matrix in suitable bases.

The **compact SVD** of $M \in \mathbb{R}^{m\times n}$ is a decomposition

$$M = U\Sigma V^T = \sum_{i=1}^{r} \sigma_i \mathrm{u}_i \otimes \mathrm{v}_i \quad \text{with } U^T U = I \text{ and } V^T V = I,$$

where r is the rank of M, U and V have **orthogonal columns**, and

$$\Sigma = \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r) \quad \text{with } \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0.$$

The number of nonzero singular values equals the **rank** of M.

The solution to the best rank-r approximation problem is readily obtained from the SVD.

Theorem (Schmidt, Mirsky, and Eckart and Young)

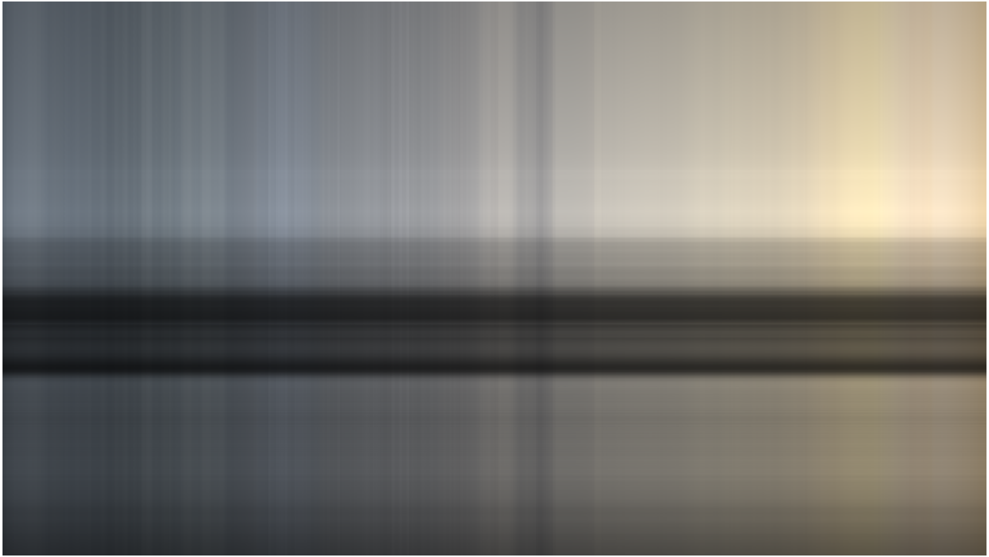
Let $A \in \mathbb{R}^{m\times n}$ have the SVD $A = U\Sigma V^T$ with $\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_r)$ and $\sigma_1 \geq \ldots \geq \sigma_r > 0$. Then,

$$\min_{\operatorname{rank}(B)\leq k} \|A - B\|_F = \left\| A - \sum_{i=1}^{k} \sigma_i \mathrm{u}_i \otimes \mathrm{v}_i \right\|_F = \sqrt{\sum_{i=k+1}^{r} \sigma_i^2}$$

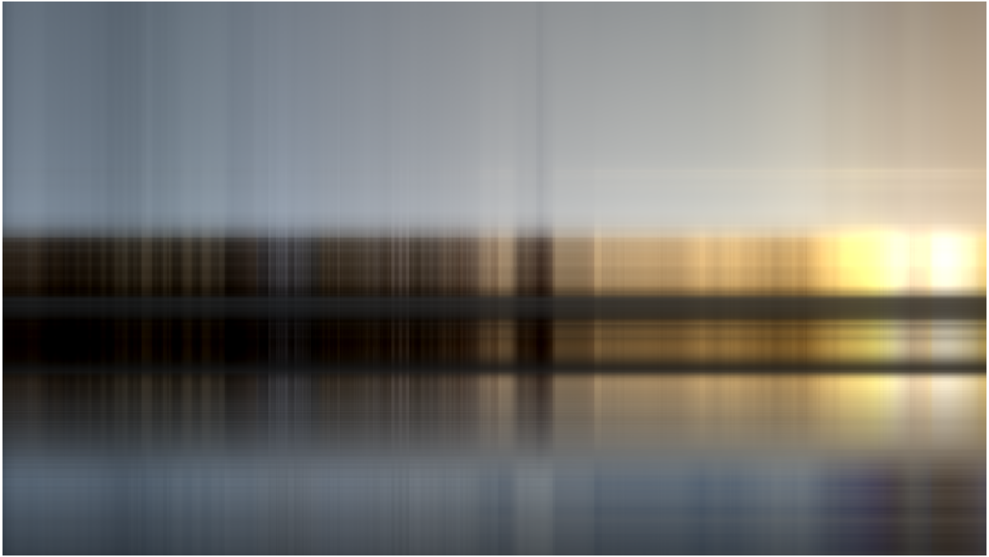
and

$$\min_{\operatorname{rank}(B)\leq k} \|A - B\|_2 = \left\| A - \sum_{i=1}^{k} \sigma_i \mathrm{u}_i \otimes \mathrm{v}_i \right\|_2 = \sigma_{k+1}$$

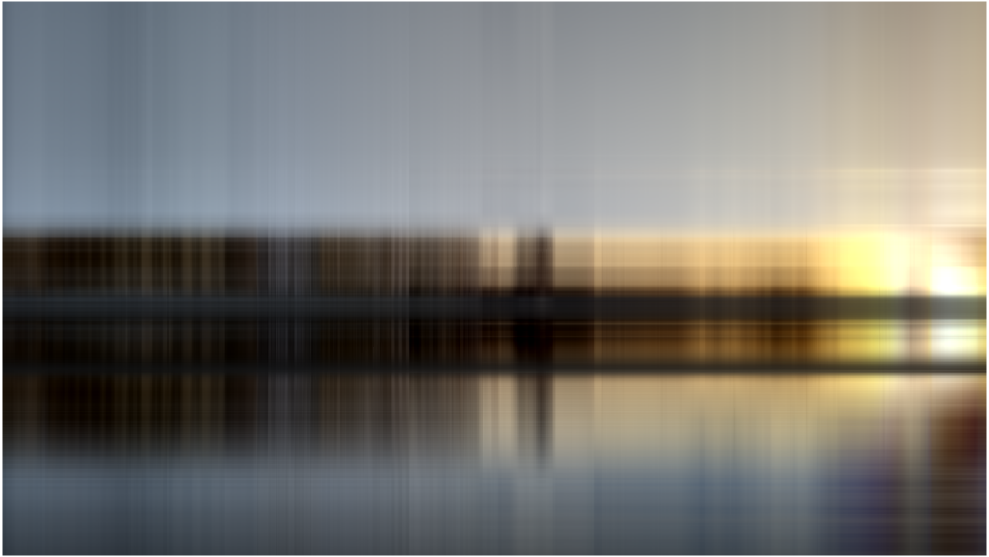
for all $1 \leq k \leq r$.



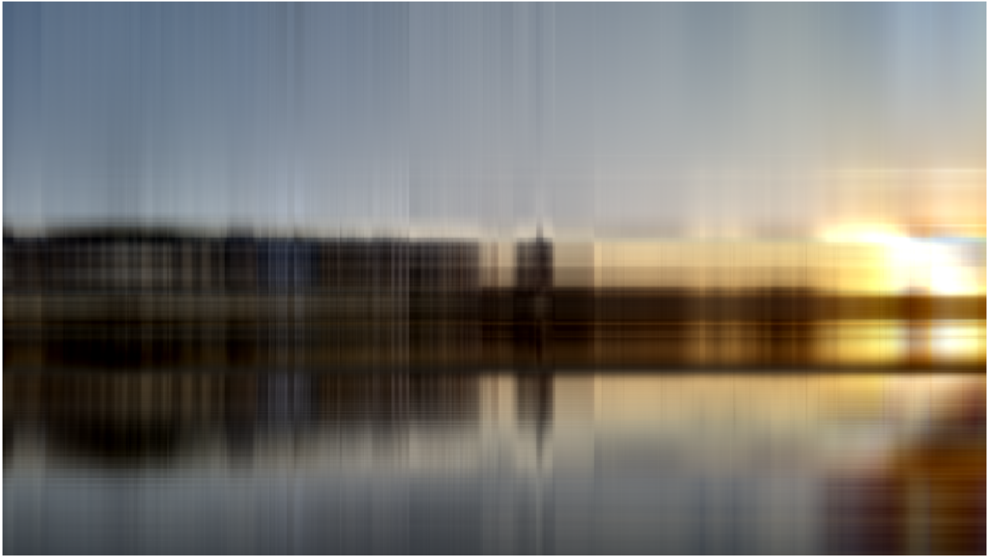
Rank 1



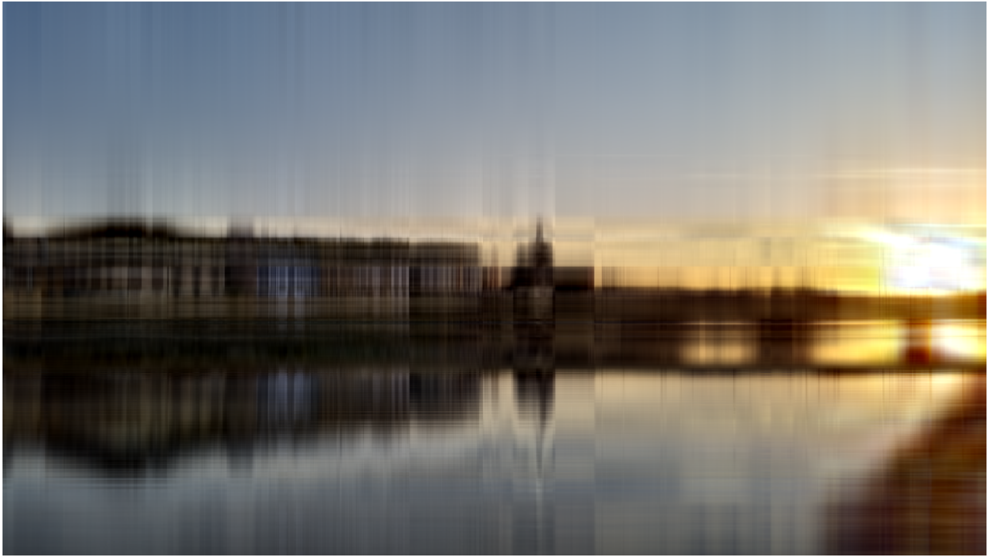
Rank 2



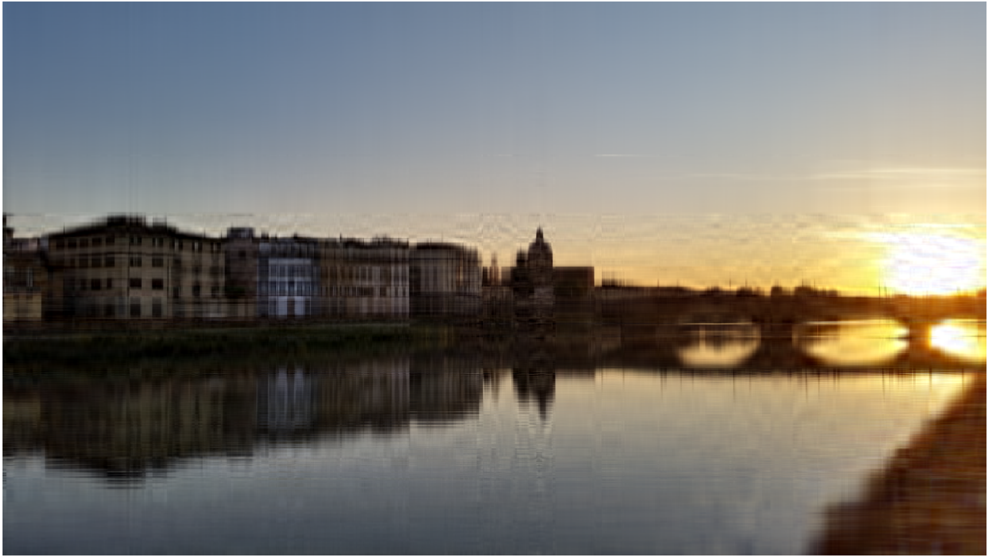
Rank 3



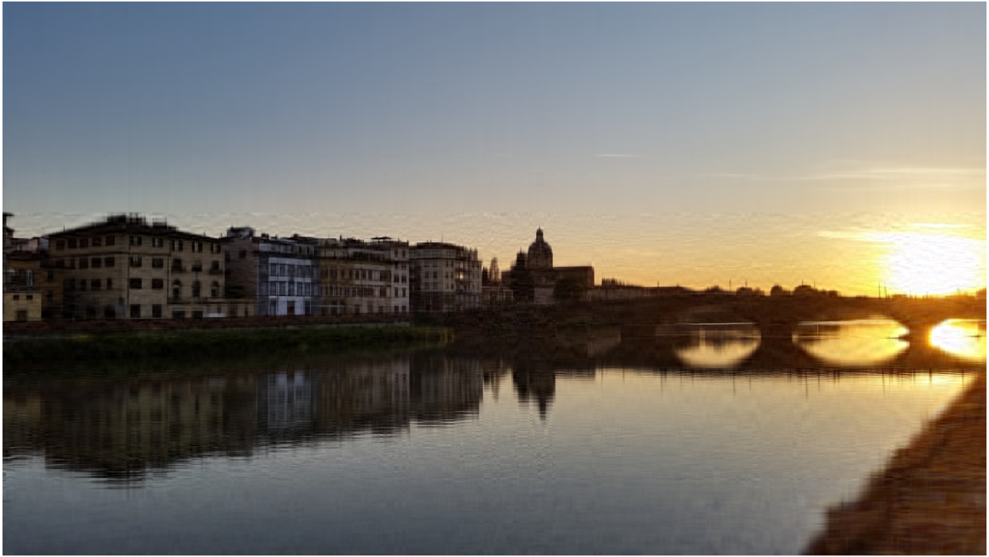
Rank 5



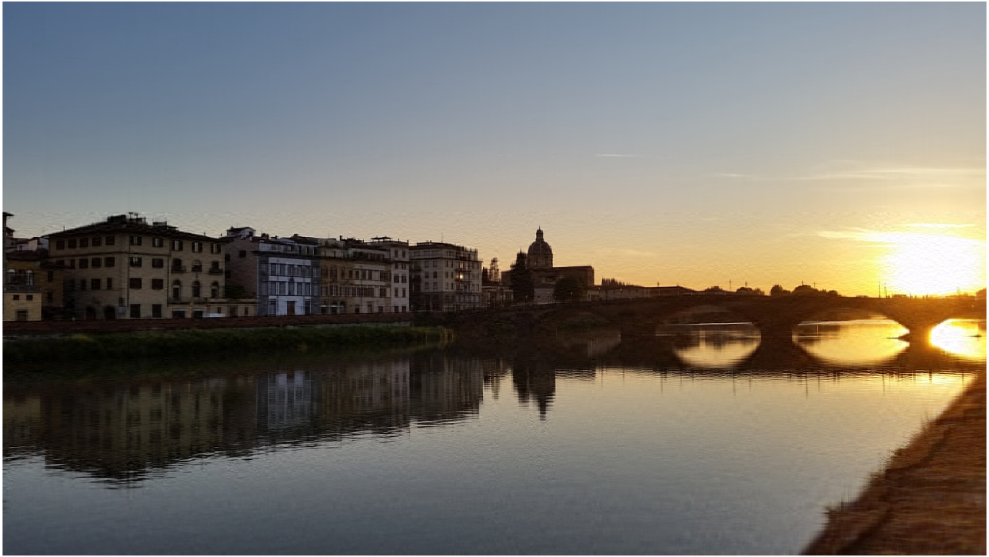
Rank 10



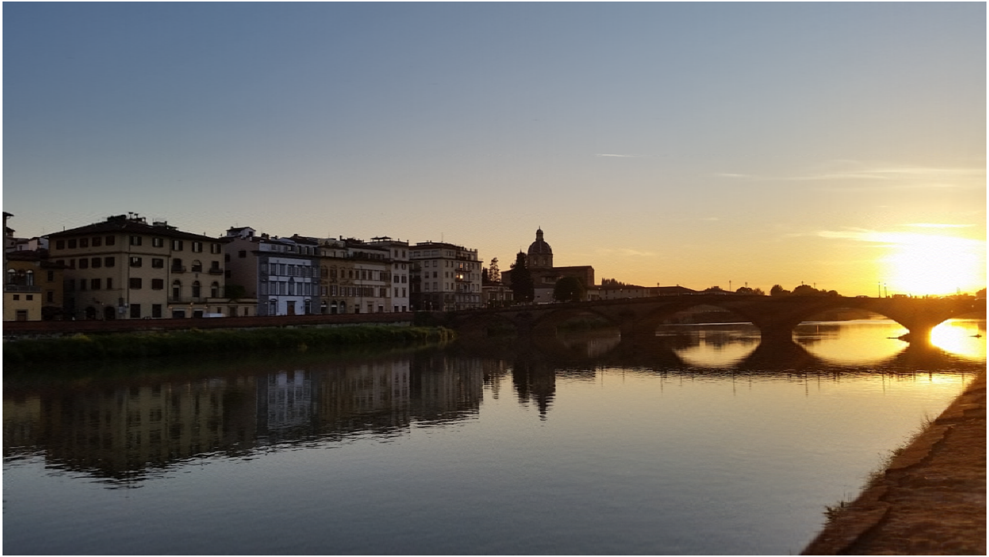
Rank 25



Rank 50



Rank 100



Rank 200

Computing the truncated SVD

The standard **numerically stable** method for computing a truncated SVD consists of

1. computing the compact SVD, and
2. truncating it by keeping only the first r terms.

The Golub–Kahan method for computing an SVD of an $m \times n$ matrix A with $m \geq n$ has **computational complexity**

$$Cmn^2 = \mathcal{O}(mn^2)$$

This roughly means that the time for computing a rank-r truncated SVD of an $n \times n$ matrix is

n	1 000	2 000	4 000	8 000	16 000	32 000
time	00:00:01	00:00:08	00:01:04	00:08:32	01:08:16	546:08:00

For data analysis applications, **full precision is rarely required**. Cheap approximations of the rank-r truncated SVD were developed based on

- simple **randomized range finders**,
- **subspace iteration** range finders,
- range finders with structured matrices, and
- **adaptive cross approximation**,
- **interpolatory decomposition**.

An overview of how to **find structure with randomness** was presented by Halko, Martinsson, and Tropp (2011).

→ I. Randomized range finder

The idea of a **randomized range finder** is as follows.

Assume that an $m \times n$ matrix A has rank r and its compact SVD is $A = USV^T$. Then, for every matrix X:

$$AX = U(SV^T X) \subset \mathrm{span}(U)$$

Moreover, if X is a random Gaussian $n \times R$ matrix with $r' \geq r$, then

$$\mathrm{span}(AX) = \mathrm{span}(U)$$

with probability 1.

Therefore, the following **randomized algorithm** produces an SVD of A:

1. choose a random $X \in \mathbb{R}^{m \times r'}$ and set $A' = AX$
2. compute a rank-r truncated SVD $A' \approx QS'W^T$
3. project onto the range of Q: $C = Q^T A$
4. compute the SVD of $C = U'SV^T$
5. set $A = (QU')SV^T$

The computational complexity of this algorithm is

$$\mathcal{O}\Big(\underbrace{mnr'}_{\text{step 1}} + \underbrace{m(r')^2}_{\text{step 2}} + \underbrace{mnr}_{\text{step 3}} + \underbrace{nr^2}_{\text{step 4}} + \underbrace{mr^2 + mnr}_{\text{step 5}}\Big) = \mathcal{O}(mnr') \ll \mathcal{O}(mn^2)$$

A simple Julia implementation may look as follows:

```
using LinearAlgebra

""" Compute the column span of A using a randomized range finder. """
function colspan_rrf(A :: Matrix, tr :: Integer)
    R = randn(size(A,2), tr+10)
    comprFact = svd(A*R)
    U = Matrix(comprFact.U[:,1:tr])
    return U
end
""" Computes a rank-r truncated SVD. """
function truncated_svd(A :: Matrix, rk :: Integer, rf = colspan_rrf)
    Q = rf(A, rk)
    C = transpose(Q)*A
    svdF = svd(C)
    return (Q*svdF.U, svdF.S, svdF.V)
end
```

→ II. Adaptive cross approximation

In adaptive cross approximation (Goreinov, Tyrtyshnikov and Zamarashkin, 1997), an **interpolating rank-1 approximation** of $A \in \mathbb{R}^{m \times n}$ is given by the "skeleton"

$$S_1 = A_{:,j^*} a_{i^*,j^*}^{-1} A_{i_*,:}$$

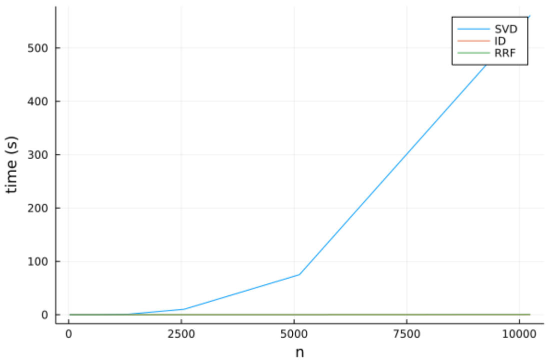
where (i^*, j^*) is the index of the largest element (in absolute value) of A.

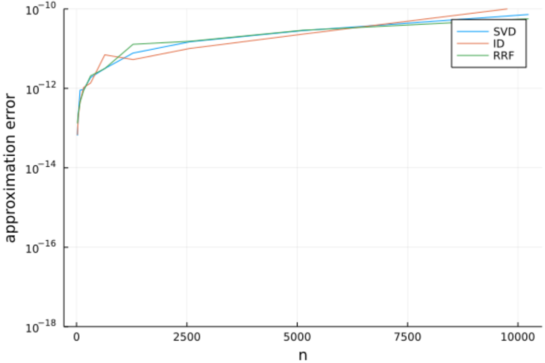
Then, we could repeat this idea, computing a rank-1 interpolating approximation of the residual $A - S_1$, resulting in S_2. Repeat this process r times to obtain the rank-r approximation

$$A \approx S_1 + S_2 + \cdots + S_r.$$

Naturally, finding the optimal (i^*, j^*) would require mn operations. This index finding is approximated by, e.g., the **rook pivoting** strategy.

Let's compare the standard rank-r truncated SVD (SVD) with a method based on the simple randomized range finder (RRF) and another one based on the interpolatory decomposition (ID). We use an exact rank-10 matrix of size $n \times n$ and truncate it to $r = 10$.





ANY QUESTIONS?
UH OH, I HAVE A QUESTION.
SHOULD I ASK IT? WHAT IF HE ALREADY ANSWERED IT? IS IT A DUMB QUESTION? AM I GOING TO EMBARRASS MYSELF IN FRONT OF EVERYONE?
MAYBE I SHOULD WAIT UNTIL AFTER THE TALK. DO I REALLY NEED TO ASK IT IN PUBLIC? BUT WHAT IF I'M NOT THE ONLY ONE? SHOULD I DO IT???
YES, CECILIA?
CAN I TAKE THE LEFT OVER COOKIES HOME?
JORGE CHAM © 2013
WWW.PHDCOMICS.COM

Overview

1. About this Masterclass (15')

2. Why tensor decompositions and their applications? (25')

3. Low-rank matrix decompositions* (35')

4. Application: analysis of energy output of solar panels (20')

5. Generalizing the SVD (10')

6. Conclusions

Application: analysis of energy output of solar panels

The rank-r truncated SVD is a **key tool in data analysis** because **data-generating processes in applications are structured**.

For example:

Application: analysis of energy output of solar panels

The rank-r truncated SVD is a **key tool in data analysis** because **data-generating processes in applications are structured**.

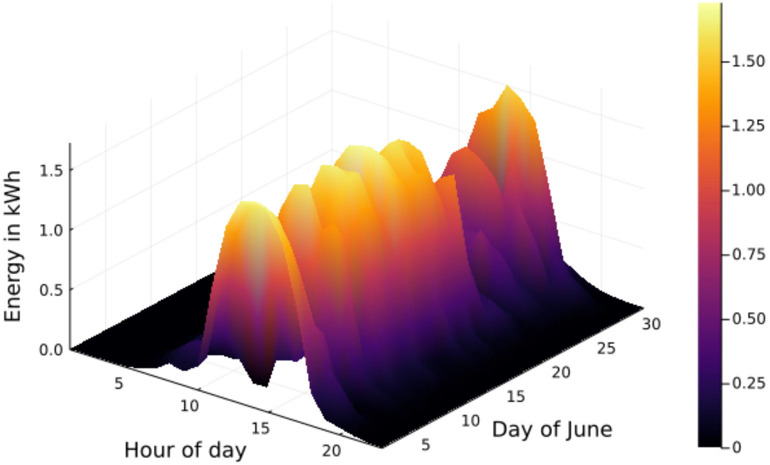
For example:

Data from several applications tends to be of low rank (Udell and Townsend, 2019).

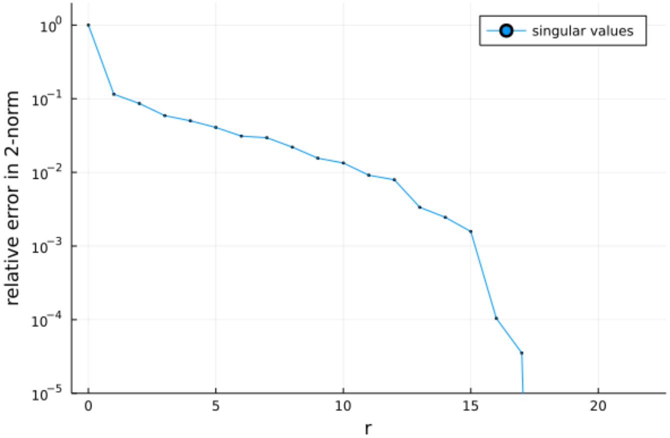
Truncated SVDs, enable us to

1. analyze the most important features of the data (**exploratory data analysis**), and
2. perform **noise reduction**,
3. **reduce the dimensionality**.

Consider the following 23×30 **data matrix** that I collected from our photovoltaic solar panels. It records the hourly energy output in kWh over the course of June 2021.



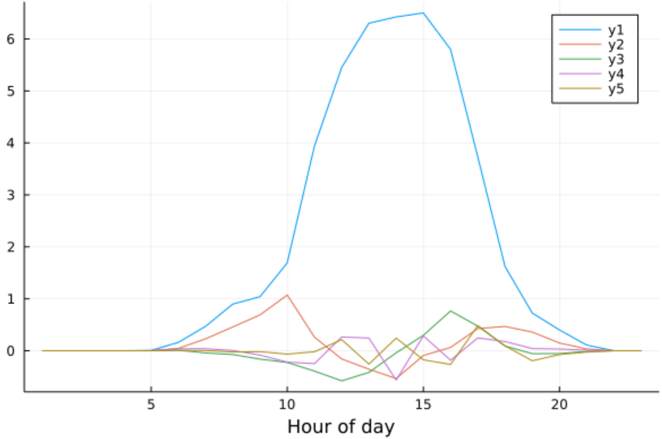
The singular values of the (normalized) matrix show a quite typical decay for many data arising in applications:



With a rank-1 approximation, 90% of the data is explained!
With a rank-10 approximation 99% of the data is explained!

1. Exploratory data analysis

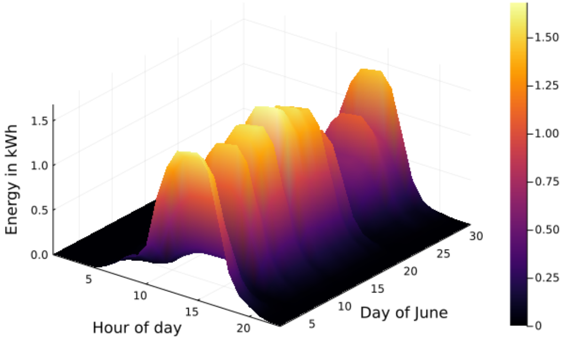
The dominant left singular vectors scaled by the corresponding singular values show the dominant behavior of the data as the hour of the day varies:



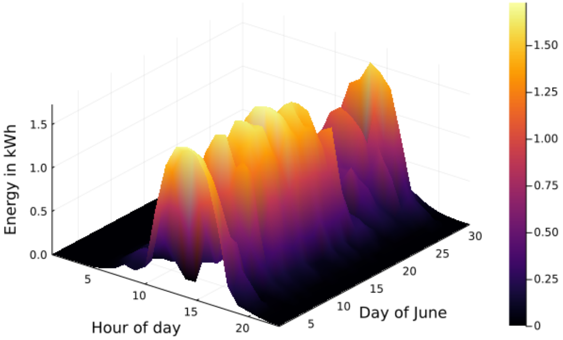
A rank-5 approximation leaves only 4% of the data unexplained.

2. Noise reduction

By taking a truncated SVD, the nondominant features of the data are removed. This has the effect of reducing noise (usually of high frequency and unstructured).



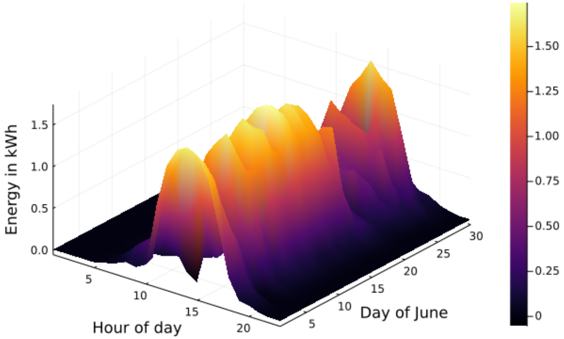
Rank 1



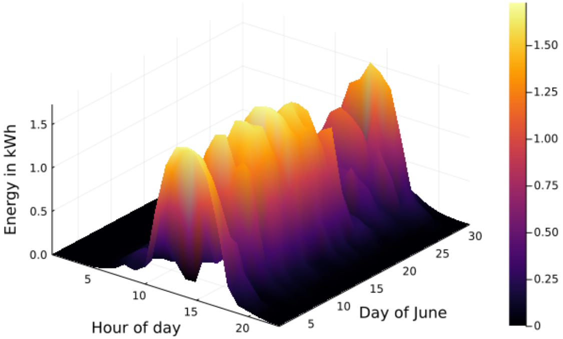
True

2. Noise reduction

By taking a truncated SVD, the nondominant features of the data are removed. This has the effect of reducing noise (usually of high frequency and unstructured).

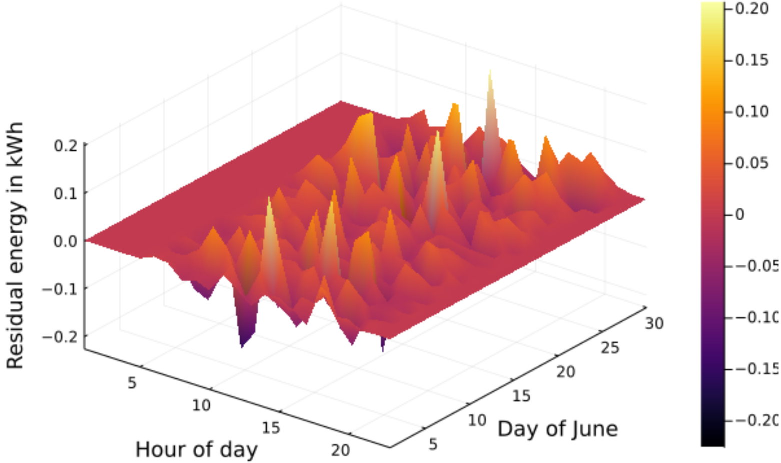


Rank 5



True

Note that the residual approximation error does not look like it contains additional structured information:



3. Dimensionality reduction

Storing a rank-r truncated SVD (U, S, V) of an $m \times n$ matrix $A = USV^T$ reduces storage from mn to $(m+n)r$ in practice.

In our case, a compression factor of

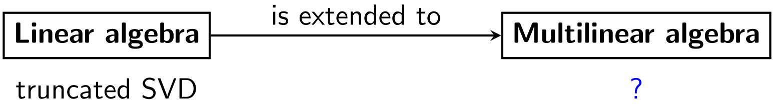
$$\frac{30 \cdot 23}{4(30+23)} = \frac{690}{212} \approx 3.25$$

ANY QUESTIONS?
UH OH, I HAVE A QUESTION.
SHOULD I ASK IT? WHAT IF HE ALREADY ANSWERED IT? IS IT A DUMB QUESTION? AM I GOING TO EMBARRASS MYSELF IN FRONT OF EVERYONE?
MAYBE I SHOULD WAIT UNTIL AFTER THE TALK. DO I REALLY NEED TO ASK IT IN PUBLIC? BUT WHAT IF I'M NOT THE ONLY ONE? SHOULD I DO IT???
JORGE CHAM © 2013
YES, CECILIA?
CAN I TAKE THE LEFT OVER COOKIES HOME?
WWW.PHDCOMICS.COM

Overview

1. About this Masterclass (15')
2. Why tensor decompositions and their applications? (25')
3. Low-rank matrix decompositions* (35')
4. Application: analysis of energy output of solar panels (20')
5. Generalizing the SVD (10')
6. Conclusions

Generalizing the SVD



Linear algebra is extended to **Multilinear algebra**

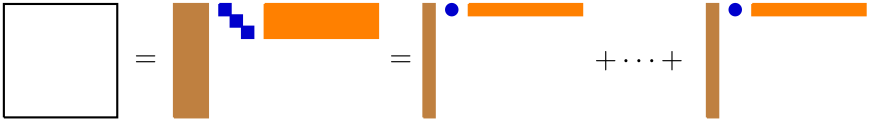
truncated SVD

?

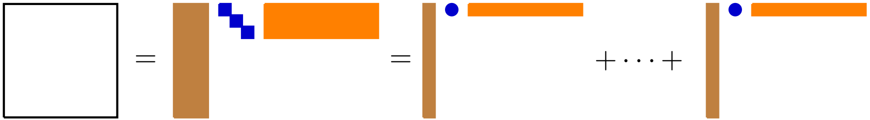
Several tensor decompositions emerged from the effort of trying to generalize the truncated SVD, each with their own use cases:

1. the **tensor rank** decomposition (Hitchcock, 1927),
2. the **Tucker** decomposition (Tucker, 1963), and
3. the **tensor trains** decomposition (Fannes, Nachtergaele, and Werner, 1992).

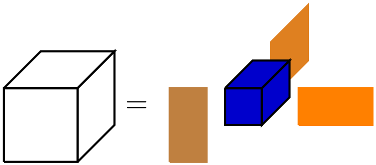
Compact singular value decomposition:



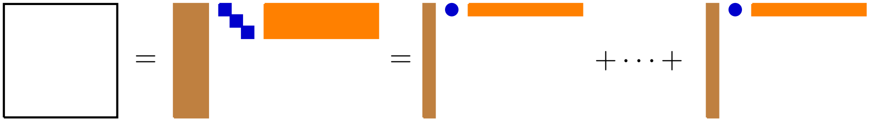
Compact singular value decomposition:



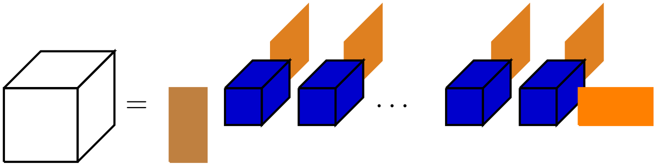
Tucker decomposition



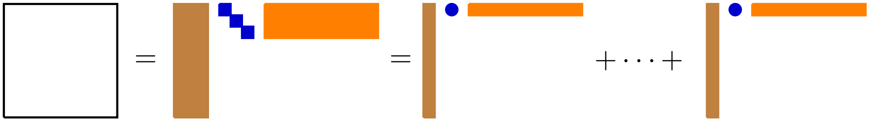
Compact singular value decomposition:



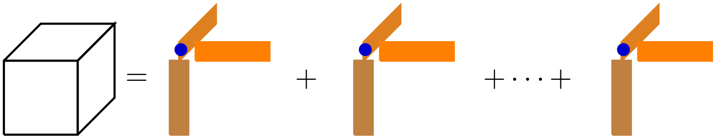
Tensor trains decomposition



Compact singular value decomposition:



Tensor rank decomposition



ANY QUESTIONS?
UH OH, I HAVE A QUESTION.
SHOULD I ASK IT? WHAT IF HE ALREADY ANSWERED IT? IS IT A DUMB QUESTION? AM I GOING TO EMBARRASS MYSELF IN FRONT OF EVERYONE?
MAYBE I SHOULD WAIT UNTIL AFTER THE TALK. DO I REALLY NEED TO ASK IT IN PUBLIC? BUT WHAT IF I'M NOT THE ONLY ONE? SHOULD I DO IT???
YES, CECILIA?
CAN I TAKE THE LEFT OVER COOKIES HOME?
JORGE CHAM © 2013
WWW.PHDCOMICS.COM

Overview

1. About this Masterclass (15')
2. Why tensor decompositions and their applications? (25')
3. Low-rank matrix decompositions* (35')
4. Application: analysis of energy output of solar panels (20')
5. Generalizing the SVD (10')
6. Conclusions

Conclusions

This is random data ... and this is real data.

References

- Demmel, *Applied Numerical Linear Algebra*, SIAM, 1997.
- Edelman and Jeong, *Fifty three matrix factorizations: A systematic approach*, arXiv:2104.08669, 2021.
- Eckart and Young, *The approximation of one matrix by another of lower rank*, Psychometrika, 1936.
- Fannes, Nachtergaele, and Werner, *Finitely correlated states on quantum spin chains*, Communications in Mathematical Physics, 1992.
- Goreinov, Tyrtyshnikov, and Zamarashkin, *A theory of pseudoskeleton approximations*, Linear Algebra and its Applications, 1997.
- Halko, Martinsson, and Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Review, 2011.
- Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM, 2002.
- Mirsky, *Symmetric gauge functions and unitarily invariant norms*, The Quarterly Journal of Mathematics, 1960.
- Schmidt, *Zur Theorie der linearen und nichtlinearen Integralgleichungen*, Mathematische Annalen, 1907.
- Stewart, *The decompositional approach to matrix computation*, Computing in Science & Engineering, 2000.
- Udell and Townsend, *Why are big data matrices approximately low rank?*, SIAM Journal on Mathematics of Data Science, 2019.